@const : 값 변경 안됨

let : 다시 선언할 수 없음.

@var 로선언하면 글로벌로 잡힌다

for(var i =0; i<=3; ~)

{ }

log(i)

=> 4 출력

@Hoisting 선언한 함수와 변수를 가장 상단에 있는 것처럼 인식...

@변수 선언문과 함수선언문 메모리 할당이 일어나고 값 할당 연산이 그 다음에 일어난다

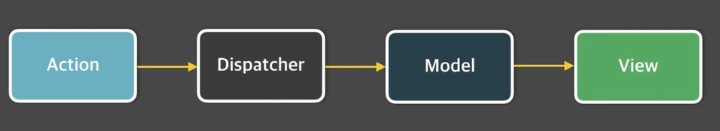
@화살표 함수 사용

@ES6 문법 스터디

스캐폴딩?

default를 넣으면 하나의 export밖에 되지않음.

Flux 패턴



state = data

getters = computed

mutations = state 값을 변경하는 이벤트 로직/methods

actions = 비동기처리 로직을 선언하는 aysnc methods

부모에서는 props만 내리고 자식에서는 이벤트만 보낸다...

action : 화면에서 발생하는 이벤트 / 사용자 입력

dispatcher : 데이터를 변경하는 방법.. 메서드

model

view

mvc 패턴의 문제점 : 모델과 뷰가 양방향으로 통신하기 때문에 추적이 어려워진다.

settimeout이나 서버api콜 같은 것을 actions에서 처리...

비동기 로직만 처리하고 데이터를 바꾸진 않음...

그러한 처리는 mutations에서함.

Vue.use(vuex) : 뷰를 사용하는 영역에 글로벌하게 사용.

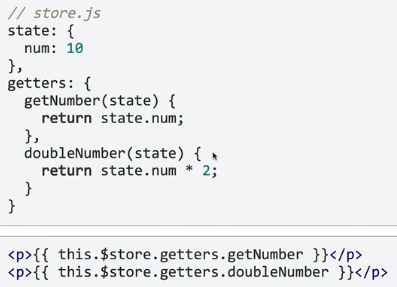
->Todo.vue에서 this.$store 접근이 가능하게한다.

Vuex를 쓰면서 달라지는 점

1.



2.



<!-- <h1>Todoit!</h1> -->

<h1>{{this.$store.state.headerText}}</h1>

//store.js

export const store = new Vuex.Store({

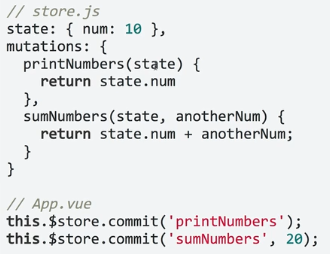
state:{

headerText : 'Todo it'

}

});

Mutations



항상 속성 메서드의 첫번째 인자로 state를 가져온다.

Commit() : mutations를 동작하게 함



commit 인자로 객체도 넘길 수도 있다.

Actions 비동기 코드 예제



State 값의 변화를 추적하기 어렵기 때문에 mutations 속성에는 동기 처리 로직만 넣는다.

this.$store.num -> mapState[‘num’]

… 쩜쩜쩜 : object spread 오퍼레이터 = 객체의 속성을 다 불러온다